

**Module: CS4001 Course work 2023-2024**

**Student Name: Ugochukwu Chikwendu**

**Student ID: 20040167**

**Weighting: 60% of module mark**

**Module Leader: Dr Sandra Fernando**

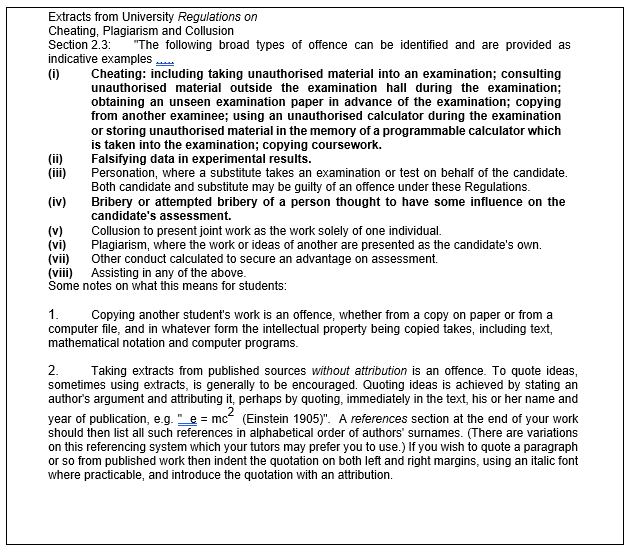
PLAGIARISM

You are reminded that there exist regulations concerning plagiarism. Extracts from these regulations are printed below. Please sign below to say that you have read and understand these extracts:

**Signature: ugo** **Date: july, 2024**

This header sheet should be attached to the work you submit. No work will be accepted without it.

Github Link : https://github.com/Ugo-Cee/Ugo-Java-coursework.git



**Content**

[Introduction 4](#_Toc172916872)

[Methods 4](#_Toc172916873)

[Gadget.java 4](#_Toc172916874)

[Mobile.java Methods 4](#_Toc172916875)

[MP3.java Methods 5](#_Toc172916876)

[Pseudocode for Button-Handling Methods 5](#_Toc172916877)

[Getting the Display Number from the GUI 5](#_Toc172916878)

[Adding a Mobile 5](#_Toc172916879)

[Adding an MP3 5](#_Toc172916880)

[Displaying All Gadgets in the ArrayList 6](#_Toc172916881)

[Making a Call 6](#_Toc172916882)

[Downloading Music 6](#_Toc172916883)

[Textboxes, Input Check with Try/Catch 6](#_Toc172916884)

[Buttons and Action Performed Methods 9](#_Toc172916885)

[Command Prompt Testing 12](#_Toc172916886)

[Error Detection & Correction 13](#_Toc172916887)

[Conclusion 14](#_Toc172916888)

[REFLECTION 14](#_Toc172916889)

[References 15](#_Toc172916890)

[Appendix 16](#_Toc172916891)

[Gadget: 16](#_Toc172916892)

[Mobile: 16](#_Toc172916893)

[MP3: 17](#_Toc172916894)

# Introduction

Java is an object-oriented programming language that uses bytecode instructions that are carried out by a stack-based virtual engine. It's one of the programming languages most commonly used in enterprise development. Java's bytecode-based design has many advantages over assembly language, one of which is that code written once can run on various platforms as long as a virtual machine implementation is available for each platform.   
The project's Java code is meant to operate a gadget store. The graphical user interface (GUI) of the shop allows users to add MP3 players and cell phones to its inventory. The classes in the application are Gadget, Mobile, MP3, and GadgetShop. Though mobile and MP3 are subcategories of devices that designate particular gadget with extra features.

**Figure 1: Class Diagram**

# 

# Methods

## Gadget.java

Constructor: Initializes a Gadget object with model, price, weight, and size properties.

getModel(): Returns the model of the gadget.

getPrice(): Returns the price of the gadget.

getWeight(): Returns the weight of the gadget.

getSize(): Returns the size of the gadget.

display(): Prints the details of the gadget including model, price, weight, and size.

## Mobile.java Methods

Constructor: Initializes a Mobile object by calling the superclass constructor and sets the calling credit.

getCallingCredit(): Returns the amount of calling credit available on the mobile.

addCredit(int credit): Adds a specified amount of calling credit if it is positive; otherwise, it prints an error message.

makeCall(String phoneNumber, int duration): Deducts the specified duration from the calling credit to make a call if enough credit is available; otherwise, it prints an error message indicating insufficient credit.

display(): Displays the details of the mobile including inherited gadget properties and the available calling credit.

## MP3.java Methods

Constructor: Initializes an MP3 object by calling the superclass constructor and sets the available memory.

getAvailableMemory(): Returns the available memory on the MP3 player.

downloadMusic(double size): Deducts the specified size from the available memory for downloading music if enough memory is available; otherwise, prints an error message indicating insufficient memory.

deleteMusic(double size): Adds the specified size back to the available memory, simulating the deletion of music.

display(): Displays the details of the MP3 player including inherited gadget properties and the available memory.

# Pseudocode for Button-Handling Methods

## Getting the Display Number from the GUI

function getDisplayNumber():

number = read text from txtModel text field

if number is valid:

return number

else:

print "Invalid input"

return null

## Adding a Mobile

function addMobile():

model = get text from txtModel

price = convert txtPrice to double

weight = convert txtWeight to integer

size = get text from txtSize

credit = convert txtCredit to integer

mobile = new Mobile(model, price, weight, size, credit)

add mobile to gadgets list

display "Mobile added"

## Adding an MP3

function addMP3():

model = get text from txtModel

price = convert txtPrice to double

weight = convert txtWeight to integer

size = get text from txtSize

memory = convert txtMemory to double

mp3 = new MP3(model, price, weight, size, memory)

add mp3 to gadgets list

display "MP3 added"

## Displaying All Gadgets in the ArrayList

function displayGadgets():

for each gadget in gadgets list:

gadget.display()

## Making a Call

function makeCall():

phoneNumber = get text from txtPhoneNumber

duration = convert txtCallDuration to integer

mobile = find mobile with model number (getDisplayNumber())

if mobile is not null:

mobile.makeCall(phoneNumber, duration)

else:

display "Mobile not found"

## Downloading Music

function downloadMusic():

size = convert txtDownloadSize to double

mp3 = find MP3 with model number (getDisplayNumber())

if mp3 is not null:

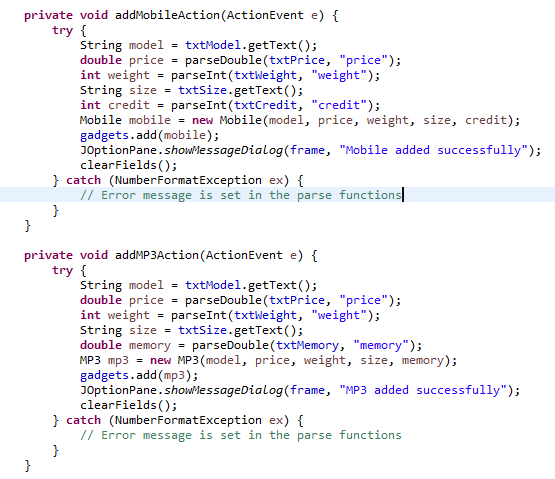
mp3. downloadMusic(size)

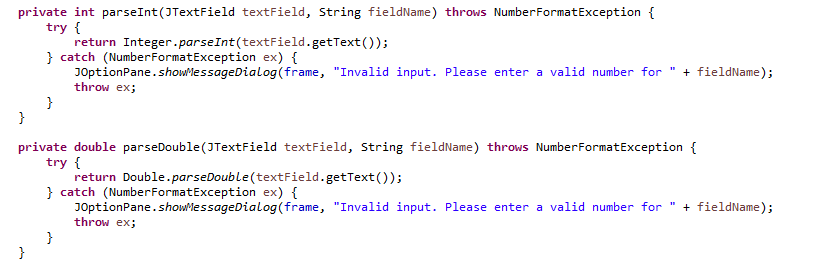
else:display "MP3 not found"

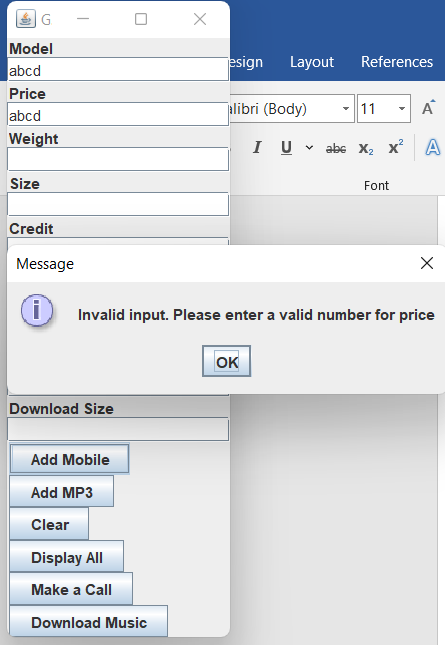
# Textboxes, Input Check with Try/Catch

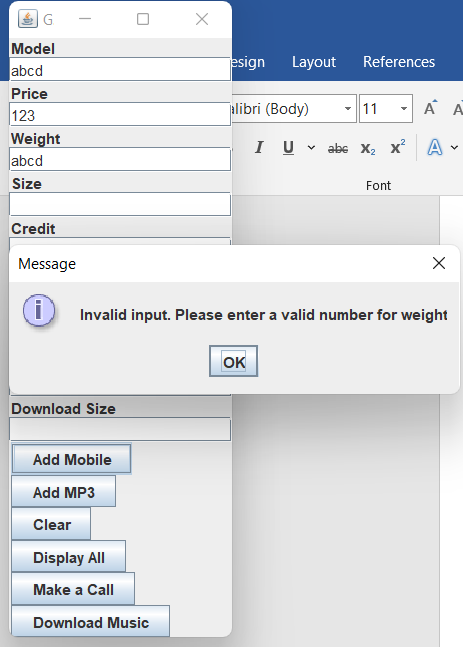
The "try-catch" technique in programming is used to encapsulate code blocks that have a chance of going wrong, like changing textbox user input into a numeric data type (Merritt, 2001). The software tries to run the code normally inside the try section. An example of an error that is caught by the catch block during execution is a NumberFormatException brought on by improper input. In this case, it manages the exception by displaying the relevant error messages, alerting the user to the wrong input, and requesting that they make the necessary corrections.

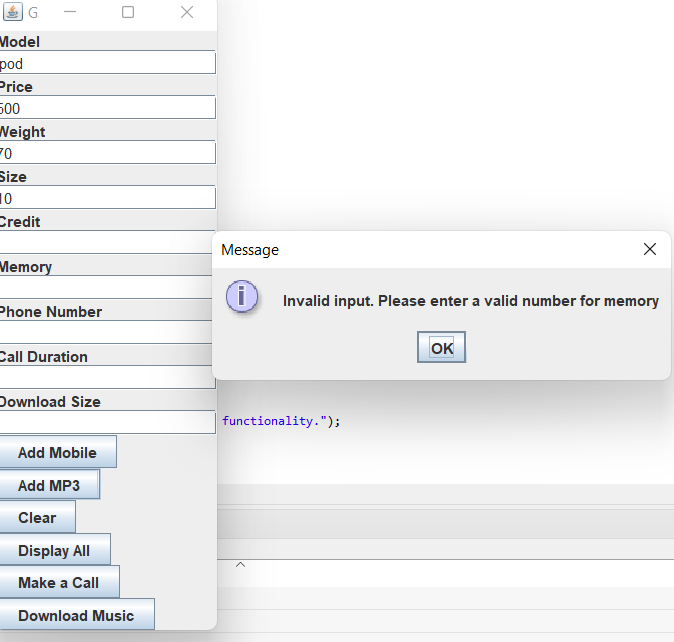
Try-catch blocks are used to prevent crashes and gently handle user input failures, hence maintaining the resilience of the application (Mehrabi et al., 2018).   
The code below has two methods, addMobileAction and addMP3Action, that deal with GUI events, such as button clicks. Every method has the same structure, with a try block containing the primary logic. These blocks employ the helper methods parseInt and parseDouble to parse user input from text fields into the proper data types, such as integers or doubles. Try-catch blocks contain these help methods, ensuring that any NumberFormatExceptions are caught and handled politely. If an exception while parsing shows incorrect data, the user displays an error message via a Joptionpane that includes the field where the problem occurred.









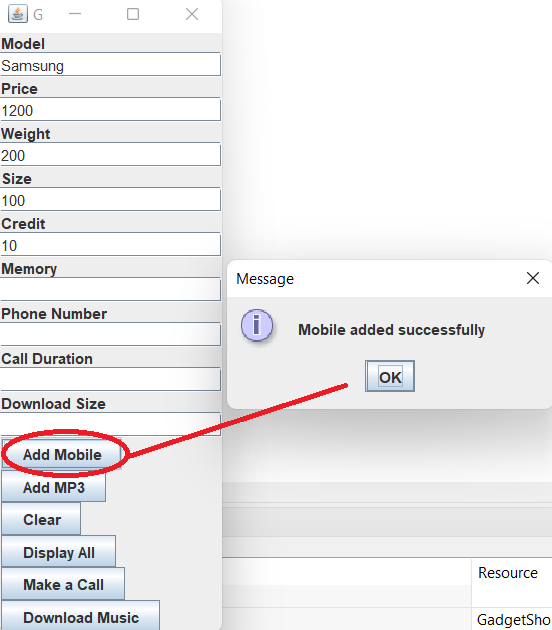


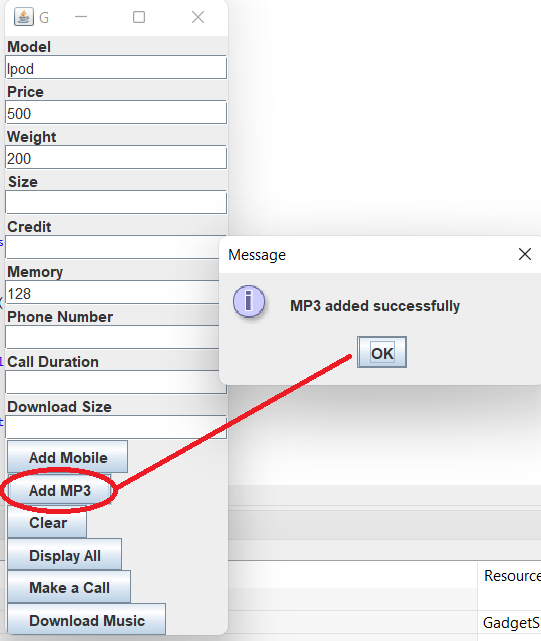
# Buttons and Action Performed Methods

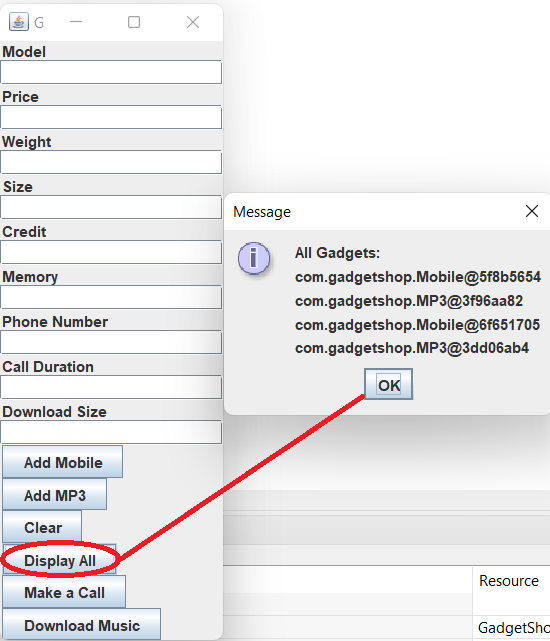
The term "graphical user interface" (GUI) refers to the visual interface that enables users to interact with the software through the use of graphical elements such buttons, text fields, and dialogues (Etheridge, 2009). The GUI makes it simple for users to enter data, initiate activities, and get replies from the program.

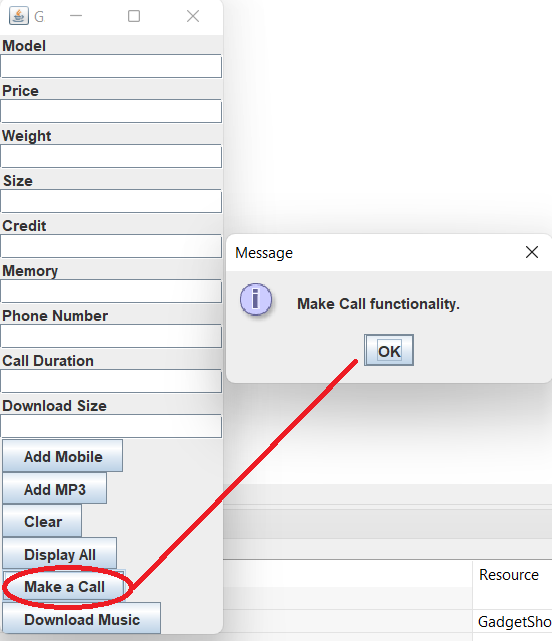
The GUI program has buttons that may be used to "Add MP3," "Add Mobile," "Display All," "Make a Call," and "Download Music." In the gadget shop programme, each button denotes a particular action the user can take. The user is prompted to enter information about the MP3 player, such as its model, price, weight, size, memory, and calling credit, using the "Add MP3" and "Add Mobile" buttons. This adds the new MP3 player or cell phone to the store's inventory as appropriate.

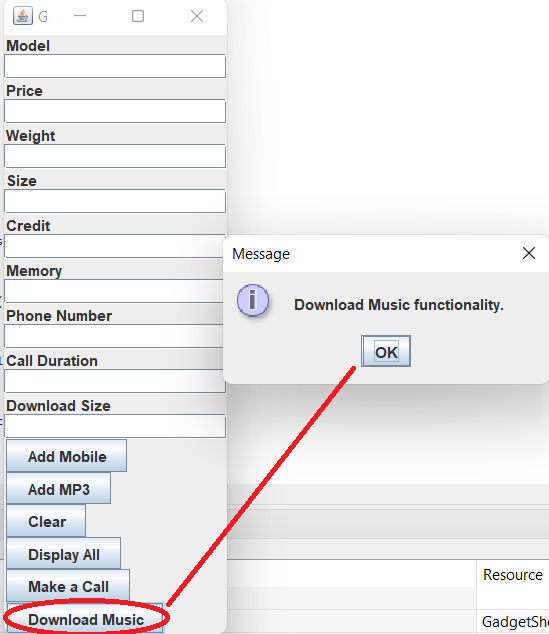
The "Display All" button allows customers to view every device that is currently listed in the inventory, giving them a thorough overview of everything that is available. Similarly, the "Download Music" and "Make a Call" buttons offer functionality that are only available on smartphones and MP3 players, respectively. Users can use these buttons to enter additional information during a call, including the phone number and length, or to enter the download size when downloading music.







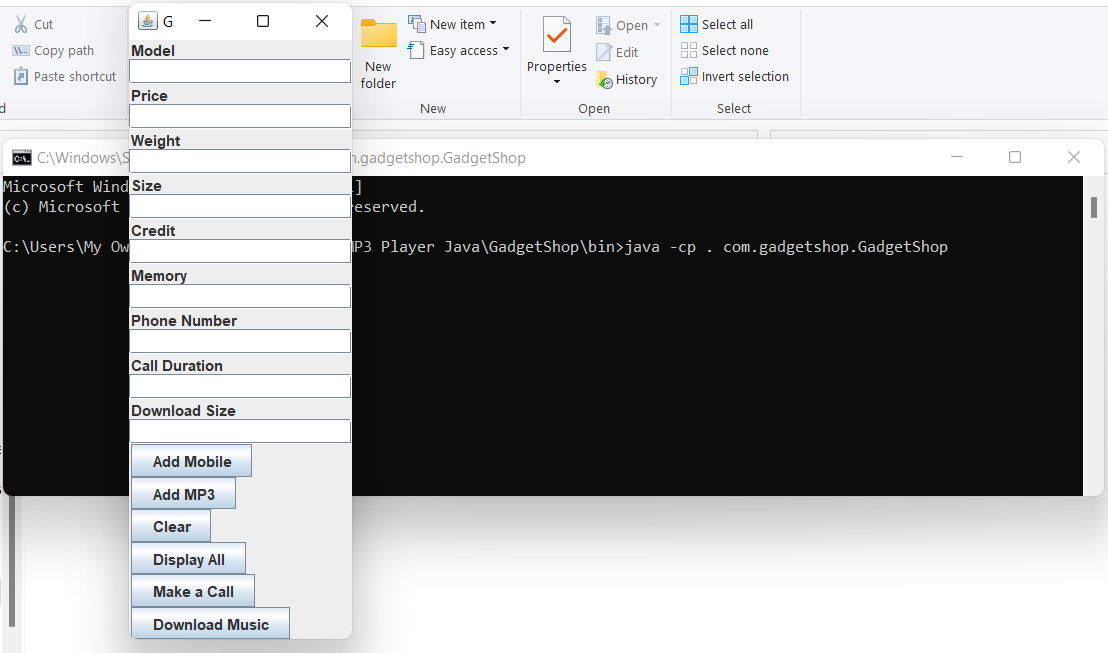




# Command Prompt Testing

Command prompt testing is a software development technique that executes the application directly from the command line interface (CLI) as opposed to an integrated development environment (IDE) or graphical user interface (GUI) (Bolepally, 2023). At the command prompt, I ran java -cp. com.gadgetshop.GadgetShop to test the program. The purpose of this command-line testing is to ensure that the software functions properly in a command-line environment and is cross-platform compatible.

It also verifies that the program may be generated and executed without an integrated development environment (IDE) with success. This command tells the Java Virtual Machine (JVM) to run the Gadget-Shop main class, which is a part of the com.gadgetshop package.   
The classpath is specified by the -cp. flag, which indicates that in order to find the classes required for execution, the current directory (.) must be included in the classpath. When the program runs, the gadget shop application is initialised and the graphical user interface (GUI) is displayed for user interaction.



# Error Detection & Correction

In particular, processes that manage passing user input are integrated with error detection and repair mechanisms. The parseInt and parseDouble methods, for example, encapsulate the conversion of text field input into integer and double data types, respectively, using try blocks. These blocks try to convert the input and deal with any NumberFormatExceptions (a warning sign for incorrect input) that could appear. When the code detects an exception, it uses JOptionPane to display an error message to the user along with the precise field (such price, weight, or memory) that has the fault. Users are able to effectively address input issues because to this instant feedback.

In addition, the function clearFields offers a temporary workaround by returning all text fields to their initial configurations, allowing users to continue after resolving input issues. Moreover, efficient error messages improve the user experience by helping users recognise and fix their errors.

# Conclusion

In conclusion, the gadget shop project shows how a Java application with a graphical user interface (GUI) for user interaction and inventory management may be implemented successfully. The creation of classes to represent devices like MP3 players and cell phones, the integration of functions to add, display, and perform operations on devices, and the construction of an intuitive graphical user interface utilising Swing components are some of the development process' major achievements. Specifically, methods for mistake detection and rectification were used to improve user experience and robustness.

However, one of the disadvantages of the gadget store project is that it relies on the Swing library to create the GUI. While web-based frameworks and JavaFX are more recent technologies, Swing makes it easier to construct cross-platform Java GUI applications, but its user interface design might not be as visually appealing or as current. This could potentially affect users' overall enjoyment and usage by making the application appear less intuitive to those used to more contemporary UI designs. In spite of this, the project exhibits useful programming strategies including modularity, error control, and user interface design in addition to meeting the necessary requirements.

# REFLECTION

It was quite difficult for me, as a novice programmer, to understand Java right away in the lecture hall. But as time goes on and after completing this class, I feel much more comfortable picking up programming in the future.   
My perspective on programming has expanded as a result of the curriculum. Even yet, I know there is still much room for improvement.  
I am grateful to all the tutors for their encouragement and patience as I am extremely optimistic about the progress I can make with their brilliant support.

# 

# References

Mehrabi, M., Giacaman, N. and Sinnen, O., 2018, May. Unobtrusive Asynchronous Exception Handling with Standard Java Try/Catch Blocks. In *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (pp. 855-864). IEEE.

Hristova, M., Misra, A., Rutter, M. and Mercuri, R., 2003. Identifying and correcting Java programming errors for introductory computer science students. *ACM Sigcse Bulletin*, *35*(1), pp.153-156.

Bolepally, A., 2023. *NAND Command Line Interface Bench test program* (Doctoral dissertation, California State University, Sacramento).

Oracle.com. (2023). Java Software. [online] Available at: <https://www.oracle.com/java> [Accessed, 25 April, 2024].

Etheridge, D., 2009. *Java: Graphical User Interfaces*. BookBoon.

Merritt, S.M., 2001. Exception handling in java: the try/catch/finally statement.

Oracle Help Center. (n.d.). JDK 22 Documentation. [online] Available at: <https://docs.oracle.com/en/java/javase/22/> [Accessed, 25 April, 2024].

# Appendix

## Gadget:

package com.gadgetshop;

public class Gadget {

    private String model;

    private double price;

    private int weight;

    private String size;

    public Gadget(String model, double price, int weight, String size) {

        this.model = model;

        this.price = price;

        this.weight = weight;

        this.size = size;

    }

    public String getModel() {

        return model;

    }

    public double getPrice() {

        return price;

    }

    public int getWeight() {

        return weight;

    }

    public String getSize() {

        return size;

    }

    public void display() {

        System.out.println("Model: " + model);

        System.out.println("Price: £" + price);

        System.out.println("Weight: " + weight + "g");

        System.out.println("Size: " + size);

    }

}

## Mobile:

package com.gadgetshop;

public class Mobile extends Gadget {

    private int callingCredit;

    public Mobile(String model, double price, int weight, String size, int callingCredit) {

        super(model, price, weight, size);

        this.callingCredit = callingCredit;

    }

    public int getCallingCredit() {

        return callingCredit;

    }

    public void addCredit(int credit) {

        if (credit > 0) {

            callingCredit += credit;

            System.out.println("Added " + credit + " minutes of credit.");

        } else {

            System.out.println("Please enter a positive amount of credit.");

        }

    }

    public void makeCall(String phoneNumber, int duration) {

        if (duration <= callingCredit) {

            callingCredit -= duration;

            System.out.println("Call made to: " + phoneNumber + " for " + duration + " minutes.");

        } else {

            System.out.println("Insufficient credit to make the call.");

        }

    }

    @Override

    public void display() {

        super.display();

        System.out.println("Calling Credit: " + callingCredit + " minutes");

    }

}

## MP3:

package com.gadgetshop;

public class MP3 extends Gadget {

    private double availableMemory;

    public MP3(String model, double price, int weight, String size, double availableMemory) {

        super(model, price, weight, size);

        this.availableMemory = availableMemory;

    }

    public double getAvailableMemory() {

        return availableMemory;

    }

    public void downloadMusic(double size) {

        if (size <= availableMemory) {

            availableMemory -= size;

            System.out.println("Downloaded " + size + "MB of music.");

        } else {

            System.out.println("Insufficient memory to download music.");

        }

    }

    public void deleteMusic(double size) {

        availableMemory += size;

        System.out.println("Deleted " + size + "MB of music, freeing up space.");

    }

    @Override

    public void display() {

        super.display();

        System.out.println("Available Memory: " + availableMemory + "MB");

    }

}

GadgetShop**:**

package com.gadgetshop;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.util.ArrayList;

public class GadgetShop {

    private ArrayList<Gadget> gadgets;

    private JFrame frame;

    private JPanel panel;

    private JTextField txtModel;

    private JTextField txtPrice;

    private JTextField txtWeight;

    private JTextField txtSize;

    private JTextField txtCredit;

    private JTextField txtMemory;

    private JTextField txtPhoneNumber;

    private JTextField txtCallDuration;

    private JTextField txtDownloadSize;

    private JButton btnAddMobile;

    private JButton btnAddMP3;

    private JButton btnClear;

    private JButton btnDisplayAll;

    private JButton btnMakeCall;

    private JButton btnDownloadMusic;

    public GadgetShop() {

        gadgets = new ArrayList<>();

        setupGUI();

    }

    private void setupGUI() {

        frame = new JFrame("Gadget Shop");

        panel = new JPanel();

        panel.setLayout(new BoxLayout(panel, BoxLayout.Y\_AXIS));

        txtModel = new JTextField(10);

        txtPrice = new JTextField(10);

        txtWeight = new JTextField(10);

        txtSize = new JTextField(10);

        txtCredit = new JTextField(10);

        txtMemory = new JTextField(10);

        txtPhoneNumber = new JTextField(10);

        txtCallDuration = new JTextField(10);

        txtDownloadSize = new JTextField(10);

        btnAddMobile = new JButton("Add Mobile");

        btnAddMP3 = new JButton("Add MP3");

        btnClear = new JButton("Clear");

        btnDisplayAll = new JButton("Display All");

        btnMakeCall = new JButton("Make a Call");

        btnDownloadMusic = new JButton("Download Music");

        // Adding action listeners to the buttons

        btnAddMobile.addActionListener(this::addMobileAction);

        btnAddMP3.addActionListener(this::addMP3Action);

        btnClear.addActionListener(e -> clearFields());

        btnDisplayAll.addActionListener(e -> displayGadgets());

        btnMakeCall.addActionListener(e -> makeCall());

        btnDownloadMusic.addActionListener(e -> downloadMusic());

        panel.add(new JLabel("Model"));

        panel.add(txtModel);

        panel.add(new JLabel("Price"));

        panel.add(txtPrice);

        panel.add(new JLabel("Weight"));

        panel.add(txtWeight);

        panel.add(new JLabel("Size"));

        panel.add(txtSize);

        panel.add(new JLabel("Credit"));

        panel.add(txtCredit);

        panel.add(new JLabel("Memory"));

        panel.add(txtMemory);

        panel.add(new JLabel("Phone Number"));

        panel.add(txtPhoneNumber);

        panel.add(new JLabel("Call Duration"));

        panel.add(txtCallDuration);

        panel.add(new JLabel("Download Size"));

        panel.add(txtDownloadSize);

        panel.add(btnAddMobile);

        panel.add(btnAddMP3);

        panel.add(btnClear);

        panel.add(btnDisplayAll);

        panel.add(btnMakeCall);

        panel.add(btnDownloadMusic);

        frame.add(panel);

        frame.pack();

        frame.setVisible(true);

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

    }

    private void addMobileAction(ActionEvent e) {

        try {

            String model = txtModel.getText();

            double price = parseDouble(txtPrice, "price");

            int weight = parseInt(txtWeight, "weight");

            String size = txtSize.getText();

            int credit = parseInt(txtCredit, "credit");

            Mobile mobile = new Mobile(model, price, weight, size, credit);

            gadgets.add(mobile);

            JOptionPane.showMessageDialog(frame, "Mobile added successfully");

            clearFields();

        } catch (NumberFormatException ex) {

            // Error message is set in the parse functions

        }

    }

    private void addMP3Action(ActionEvent e) {

        try {

            String model = txtModel.getText();

            double price = parseDouble(txtPrice, "price");

            int weight = parseInt(txtWeight, "weight");

            String size = txtSize.getText();

            double memory = parseDouble(txtMemory, "memory");

            MP3 mp3 = new MP3(model, price, weight, size, memory);

            gadgets.add(mp3);

            JOptionPane.showMessageDialog(frame, "MP3 added successfully");

            clearFields();

        } catch (NumberFormatException ex) {

            // Error message is set in the parse functions

        }

    }

    private int parseInt(JTextField textField, String fieldName) throws NumberFormatException {

        try {

            return Integer.parseInt(textField.getText());

        } catch (NumberFormatException ex) {

            JOptionPane.showMessageDialog(frame, "Invalid input. Please enter a valid number for " + fieldName);

            throw ex;

        }

    }

    private double parseDouble(JTextField textField, String fieldName) throws NumberFormatException {

        try {

            return Double.parseDouble(textField.getText());

        } catch (NumberFormatException ex) {

            JOptionPane.showMessageDialog(frame, "Invalid input. Please enter a valid number for " + fieldName);

            throw ex;

        }

    }

    private void clearFields() {

        txtModel.setText("");

        txtPrice.setText("");

        txtWeight.setText("");

        txtSize.setText("");

        txtCredit.setText("");

        txtMemory.setText("");

        txtPhoneNumber.setText("");

        txtCallDuration.setText("");

        txtDownloadSize.setText("");

    }

    private void displayGadgets() {

        StringBuilder displayText = new StringBuilder("All Gadgets:\n");

        for (Gadget gadget : gadgets) {

            displayText.append(gadget.toString()).append("\n");

        }

        JOptionPane.showMessageDialog(frame, displayText.toString());

    }

    private void makeCall() {

        JOptionPane.showMessageDialog(frame, "Make Call functionality.");

    }

    private void downloadMusic() {

        JOptionPane.showMessageDialog(frame, "Download Music functionality.");

    }

    public static void main(String[] args) {

        new GadgetShop();

    }

}